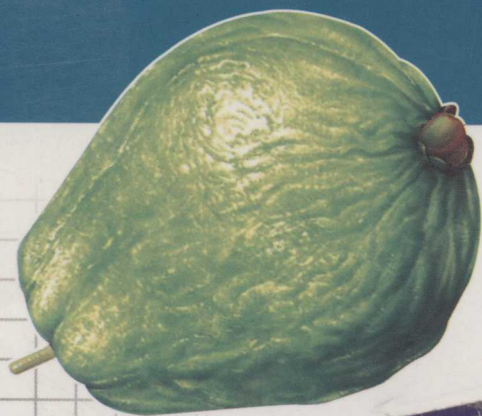


大学计算机教育国外著名教材系列

(影印版)

THE ART OF ASSEMBLY LANGUAGE

汇编语言艺术



Randall Hyde 著



清华大学出版社

417237

大学计算机教育国外著名教材系列(影印版)

出版说明

The Art of Assembly Language

汇编语言艺术

Randall Hyde 著

清华大学出版社

北京

Randall Hyde

The Art of Assembly Language

EISBN: 1-886411-97-2

Copyright © 2003 by No Starch Press

Authorized English language edition jointly published by No Starch Press and Tsinghua University Press.

This edition is authorized for sale only to the educational and training institutions, and within the territory of the People's Republic of China (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由清华大学出版社和美国 No Starch Press 合作出版。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾地区)针对教育及培训机构之销售。未经许可之出口,视为违反著作权法,将受法律之制裁。

未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号 图字:01-2005-0474

版权所有,翻印必究。举报电话:010-62782989 13501256678 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用清华大学核研院专有核径迹膜防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

汇编语言艺术 = The Art of Assembly Language / 海德(Hyde, R.) 著. —影印本. —北京:清华大学出版社, 2005.3

(大学计算机教育国外著名教材系列)

ISBN 7-302-10435-2

I. 汇… II. 海… III. 汇编语言—程序设计—高等学校—教材—英文 IV. TP313

中国版本图书馆 CIP 数据核字(2005)第 008526 号

出版者:清华大学出版社

<http://www.tup.com.cn>

社总机:010-62770175

地址:北京清华大学学研大厦

邮编:100084

客户服务:010-62776969

印刷者:清华大学印刷厂

装订者:北京鑫海金澳印刷厂

发行者:新华书店总店北京发行所

开本:185×230 印张:57

版次:2005年3月第1版 2005年3月第1次印刷

书号:ISBN 7-302-10435-2/TP·7087

印数:1~3000

定价:88.00元

ACKNOWLEDGMENTS

出版说明

进入 21 世纪, 世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才, 谁就能在竞争中取得优势。高等教育, 作为培养高素质人才的事业, 必然受到高度重视。目前我国高等教育的教材更新较慢, 为了加快教材的更新频率, 教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始, 与国外著名出版公司合作, 影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书, 受到国内读者的欢迎和支持。跨入 21 世纪, 我们本着为我国高等教育教材建设服务的初衷, 在已有的基础上, 进一步扩大选题内容, 改变图书开本尺寸, 一如既往地请有关专家挑选适用于我国高等本科及研究生计算机教育的国外经典教材或著名教材, 组成本套“大学计算机教育国外著名教材系列(影印版)”, 以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材, 以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好, 更适合高校师生的需要。

清华大学出版社

清华大学出版社

二〇〇四年四月二十日

序^①

汇编语言是计算机提供给用户的一种面向机器的编程语言，这种语言可以最大限度地利用计算机硬件特性并能通过汇编指令直接控制机器硬件，因此，利用汇编语言可以编写出在时间和空间上最具效率的程序。

要想很好地学习、掌握汇编语言，一本好书是必不可少的。《汇编语言艺术》就是近年来出现的一本好书。这本书以 80x86 系列机为背景，通过大量的程序代码详细地介绍了 80x86 汇编语言的基础知识，特别是汇编语言的编程方法和技巧。作者将多年来的开发和教学经验融合在大量的编程实例中，读者通过本书能快速地学会汇编语言程序设计，掌握其中的编程技巧，并能从一开始就养成良好的编程风格，从而实现从初学者到高级编程人员的过渡。

本书的作者 Randall Hyde 在大学中教授汇编语言十多年，并且开发了好几个商用软件，具有丰富的汇编语言教学和开发经验。在本书中作者还介绍了一种高级汇编语言 HLA (High Level Assembly)，最开始 HLA 是加州大学的教授用来讲解汇编语言编程和机器组织的一个工具，经过几年的发展，如今已经成为了资深汇编语言程序员编写可读性强、功能强大的汇编语言代码的开发平台。它的变量声明、过程声明、过程调用等都使用与高级语言类似的语法，同时还可以使用函数库。如果读者熟悉像 C/C++、Pascal/Delphi、Java 或者 VB 这样的高级语言，就会发现使用 HLA 既像高级语言一样方便，又保持了汇编语言的高效率。

本书的英文书名为“The Art of Assembly Language”。在以英语为母语的国家中，人们习惯于把一切比较需要技术的、神奇的、难以用机械的方式进行重复的东西称之为“Art”。我认为正是由于汇编语言所具有的丰富而又灵活的功能，才使程序员在程序设计中能充分发挥自己的编程技巧，就像艺术家一样，程序员也可创作出“精美”的程序来。

要说本书不足之处，我想可能就是它的“大部头”，书中有的内容写得过细，会使读者觉得有点“啰嗦”。另外，学习本书内容，需要配备 HLA 软件，但令人欣慰的是，HLA 软件可以免费下载。

很高兴看到，清华大学出版社及时引进该书的中文版和英文版，其中文版是由两位计算机博士翻译和审校的，他们较准确地把握和传递了全书的精髓。我相信，这本书一定会吸引广大专业工作者以及年青的程序设计爱好者。

清华大学教授 温冬婵

二零零四年十二月

^① 编译和运行本书程序的软件可以从 <http://www.tupwk.com.cn> 站点的“下载页面”下载。最新版本的软件还可以从 <http://webster.cs.ucr.edu> 站点下载。

ACKNOWLEDGMENTS

CONTENTS IN DETAIL

This book has literally taken over a decade to create. It started out as “How to Program the IBM PC, Using 8088 Assembly Language” way back in 1989. I originally wrote this book for the students in my assembly language course at Cal Poly Pomona and UC Riverside. Over the years, hundreds of students have made small and large contributions (it’s amazing how a little extra credit can motivate some students). I’ve also received thousands of comments via the Internet after placing an early, 16-bit edition of this book on my website at UC Riverside. I owe everyone who has contributed to this effort my gratitude.

I would also like to specifically thank Mary Phillips, who spent several months helping me proofread much of the 16-bit edition upon which I’ve based this book. Mary is a wonderful person and a great friend.

I also owe a deep debt of gratitude to William Pollock at No Starch Press, who rescued this book from obscurity. He is the one responsible for convincing me to spend some time beating on this book to create a publishable entity from it. I would also like to thank Karol Jurado for shepherding this project from its inception — it’s been a long, hard road. Thanks, Karol.

1.1 Chapter Overview	22
1.2 The Assembly Language	23
1.3 Running Your Program	24
1.4 Some Basic Instructions	25
1.5 Boolean Values	26
1.6 Character Values	27
1.7 An Introduction to the HLA Standard Library	28
1.7.1 The <code>HLA</code> Directive	29
1.7.2 The <code>INCLUDE</code> Directive	30
1.7.3 The <code>USE32</code> Directive	31
1.7.4 The <code>WHILE...ENDWHILE</code> Statement	32
1.7.5 The <code>FOR...ENDFOR</code> Statement	33
1.7.6 The <code>REPEAT...UNTIL</code> Statement	34
1.7.7 The <code>BREAK</code> and <code>BREAKIF</code> Statements	35
1.7.8 The <code>FOREVER...ENDFOR</code> Statement	36
1.7.9 The <code>TRY...EXCEPTION...ENDTRY</code> Statement	37
1.10 Introduction to the HLA Standard Library	38
1.10.1 Predefined Constants in the <code>STDIO</code> Module	39
1.10.2 Standard In and Standard Out	40
1.10.3 The <code>stdout.newln</code> Routine	41
1.10.4 The <code>stdout.putX</code> Routine	42
1.10.5 The <code>stdout.putXSize</code> Routines	43
1.10.6 The <code>stdout.put</code> Routine	44
1.10.7 The <code>stdin.get</code> Routine	45
1.10.8 The <code>stdin.getX</code> Routines	46
1.10.9 The <code>stdin.readln</code> and <code>stdin.flushput</code> Routines	47
1.10.10 The <code>stdin.get</code> Routine	48
1.11 Additional Details About <code>TRY...ENDTRY</code>	49
1.11.1 Nesting <code>TRY...ENDTRY</code> Statements	50
1.11.2 The <code>UNPROTECTED</code> Clause in a <code>TRY...ENDTRY</code> Statement	51
1.11.3 The <code>ANYEXCEPTION</code> Clause in a <code>TRY...ENDTRY</code> Statement	52
1.11.4 Registers and the <code>TRY...ENDTRY</code> Statement	53
1.12 High Level Assembly Language vs. Low Level Assembly	54
1.13 For More Information	55

CONTENTS IN DETAIL

1

HELLO, WORLD OF ASSEMBLY LANGUAGE

1.1	Chapter Overview	1
1.2	The Anatomy of an HLA Program	2
1.3	Running Your First HLA Program	4
1.4	Some Basic HLA Data Declarations	5
1.5	Boolean Values	7
1.6	Character Values	8
1.7	An Introduction to the Intel 80x86 CPU Family	8
1.7.1	The Memory Subsystem	11
1.8	Some Basic Machine Instructions	14
1.9	Some Basic HLA Control Structures	18
1.9.1	Boolean Expressions in HLA Statements	18
1.9.2	The HLA IF..THEN..ELSEIF..ELSE..ENDIF Statement	20
1.9.3	Conjunction, Disjunction, and Negation in Boolean Expressions	22
1.9.4	The WHILE..ENDWHILE Statement	25
1.9.5	The FOR..ENDFOR Statement	25
1.9.6	The REPEAT..UNTIL Statement	26
1.9.7	The BREAK and BREAKIF Statements	27
1.9.8	The FOREVER..ENDFOR Statement	28
1.9.9	The TRY..EXCEPTION..ENDTRY Statement	28
1.10	Introduction to the HLA Standard Library	32
1.10.1	Predefined Constants in the STDIO Module	33
1.10.2	Standard In and Standard Out	34
1.10.3	The stdout.newLn Routine	34
1.10.4	The stdout.putiX Routines	34
1.10.5	The stdout.putiXSize Routines	35
1.10.6	The stdout.put Routine	36
1.10.7	The stdin.getc Routine	38
1.10.8	The stdin.getiX Routines	39
1.10.9	The stdin.readLn and stdin.flushInput Routines	40
1.10.10	The stdin.get Routine	41
1.11	Additional Details About TRY..ENDTRY	42
1.11.1	Nesting TRY..ENDTRY Statements	43
1.11.2	The UNPROTECTED Clause in a TRY..ENDTRY Statement	45
1.11.3	The ANYEXCEPTION Clause in a TRY..ENDTRY Statement	48
1.11.4	Registers and the TRY..ENDTRY Statement	48
1.12	High Level Assembly Language vs. Low Level Assembly	50
1.13	For More Information	51

2

DATA REPRESENTATION

2.1	Chapter Overview	53
2.2	Numbering Systems	54
2.2.1	A Review of the Decimal System	54
2.2.2	The Binary Numbering System	54
2.2.3	Binary Formats	56
2.3	The Hexadecimal Numbering System	57
2.4	Data Organization	59
2.4.1	Bits	59
2.4.2	Nibbles	60
2.4.3	Bytes	61
2.4.4	Words	62
2.4.5	Double Words	63
2.4.6	Quad Words and Long Words	64
2.5	Arithmetic Operations on Binary and Hexadecimal Numbers	65
2.6	A Note About Numbers vs. Representation	66
2.7	Logical Operations on Bits	69
2.8	Logical Operations on Binary Numbers and Bit Strings	71
2.9	Signed and Unsigned Numbers	73
2.10	Sign Extension, Zero Extension, Contraction, and Saturation	78
2.11	Shifts and Rotates	82
2.12	Bit Fields and Packed Data	87
2.13	An Introduction to Floating Point Arithmetic	92
2.13.1	IEEE Floating Point Formats	95
2.13.2	HLA Support for Floating Point Values	99
2.14	Binary Coded Decimal (BCD) Representation	102
2.15	Characters	104
2.15.1	The ASCII Character Encoding	104
2.15.2	HLA Support for ASCII Characters	108
2.16	The Unicode Character Set	112
2.17	For More Information	113

3

MEMORY ACCESS AND ORGANIZATION

3.1	Chapter Overview	115
3.2	The 80x86 Addressing Modes	115
3.2.1	80x86 Register Addressing Modes	116
3.2.2	80x86 32-Bit Memory Addressing Modes	117
3.3	Run-Time Memory Organization	124
3.3.1	The Code Section	125
3.3.2	The Static Sections	127
3.3.3	The Read-Only Data Section	128
3.3.4	The Storage Section	129
3.3.5	The @NOSTORAGE Attribute	129
3.3.6	The Var Section	130
3.3.7	Organization of Declaration Sections Within Your Programs	131

3.4	How HLA Allocates Memory for Variables	132
3.5	HLA Support for Data Alignment	133
3.6	Address Expressions	136
3.7	Type Coercion	139
3.8	Register Type Coercion	141
3.9	The Stack Segment and the PUSH and POP Instructions	142
3.9.1	The Basic PUSH Instruction	142
3.9.2	The Basic POP Instruction	144
3.9.3	Preserving Registers with the PUSH and POP Instructions	146
3.9.4	The Stack Is a LIFO Data Structure	146
3.9.5	Other PUSH and POP Instructions	149
3.9.6	Removing Data from the Stack Without Popping It	150
3.9.7	Accessing Data You've Pushed on the Stack Without Popping It	153
3.10	Dynamic Memory Allocation and the Heap Segment	154
3.11	The INC and DEC Instructions	159
3.12	Obtaining the Address of a Memory Object	159
3.13	For More Information	160

4

CONSTANTS, VARIABLES, AND DATA TYPES

4.1	Chapter Overview	161
4.2	Some Additional Instructions: INTMUL, BOUND, INTO	162
4.3	The TBYTE Data Types	166
4.4	HLA Constant and Value Declarations	167
4.4.1	Constant Types	170
4.4.2	String and Character Literal Constants	171
4.4.3	String and Text Constants in the CONST Section	174
4.4.4	Constant Expressions	175
4.4.5	Multiple CONST Sections and Their Order in an HLA Program	178
4.4.6	The HLA VAL Section	178
4.4.7	Modifying VAL Objects at Arbitrary Points in Your Programs	179
4.5	The HLA TYPE Section	180
4.6	ENUM and HLA Enumerated Data Types	181
4.7	Pointer Data Types	182
4.7.1	Using Pointers in Assembly Language	184
4.7.2	Declaring Pointers in HLA	185
4.7.3	Pointer Constants and Pointer Constant Expressions	185
4.7.4	Pointer Variables and Dynamic Memory Allocation	187
4.7.5	Common Pointer Problems	188
4.8	The HLA Standard Library CHARS.HHF Module	192
4.9	Composite Data Types	195
4.10	Character Strings	195
4.11	HLA Strings	198
4.12	Accessing the Characters Within a String	204
4.13	The HLA String Module and Other String-Related Routines	206
4.14	In-Memory Conversions	219
4.15	Character Sets	220
4.16	Character Set Implementation in HLA	221
4.17	HLA Character Set Constants and Character Set Expressions	223

4.18	The IN Operator in HLA HLL Boolean Expressions	224
4.19	Character Set Support in the HLA Standard Library	225
4.20	Using Character Sets in Your HLA Programs	229
4.21	Arrays	230
4.22	Declaring Arrays in Your HLA Programs	231
4.23	HLA Array Constants	232
4.24	Accessing Elements of a Single Dimension Array	233
4.24.1	Sorting an Array of Values	235
4.25	Multidimensional Arrays	237
4.25.1	Row Major Ordering	238
4.25.2	Column Major Ordering	242
4.26	Allocating Storage for Multidimensional Arrays	243
4.27	Accessing Multidimensional Array Elements in Assembly Language	245
4.28	Large Arrays and MASM (Windows Programmers Only)	246
4.29	Records	247
4.30	Record Constants	249
4.31	Arrays of Records	250
4.32	Arrays/Records as Record Fields	251
4.33	Controlling Field Offsets Within a Record	255
4.34	Aligning Fields Within a Record	256
4.35	Pointers to Records	257
4.36	Unions	259
4.37	Anonymous Unions	262
4.38	Variant Types	262
4.39	Union Constants	263
4.40	Namespaces	264
4.41	Dynamic Arrays in Assembly Language	268
4.42	HLA Standard Library Array Support	270
4.43	For More Information	273

5

PROCEDURES AND UNITS

5.1	Chapter Overview	275
5.2	Procedures	276
5.3	Saving the State of the Machine	278
5.4	Prematurely Returning from a Procedure	282
5.5	Local Variables	283
5.6	Other Local and Global Symbol Types	289
5.7	Parameters	289
5.7.1	Pass by Value	290
5.7.2	Pass by Reference	293
5.8	Functions and Function Results	296
5.8.1	Returning Function Results	297
5.8.2	Instruction Composition in HLA	298
5.8.3	The HLA @RETURNS Option in Procedures	301
5.9	Recursion	303
5.10	Forward Procedures	307
5.11	Low Level Procedures and the CALL Instruction	308
5.12	Procedures and the Stack	311

5.13	Activation Records	314
5.14	The Standard Entry Sequence	317
5.15	The Standard Exit Sequence	318
5.16	Low Level Implementation of Automatic (Local) Variables	320
5.17	Low Level Parameter Implementation	322
5.17.1	Passing Parameters in Registers	322
5.17.2	Passing Parameters in the Code Stream	325
5.17.3	Passing Parameters on the Stack	328
5.18	Procedure Pointers	350
5.19	Procedure Parameters	354
5.20	Untyped Reference Parameters	355
5.21	Managing Large Programs	356
5.22	The #INCLUDE Directive	357
5.23	Ignoring Duplicate #INCLUDE Operations	358
5.24	UNITs and the EXTERNAL Directive	359
5.24.1	Behavior of the EXTERNAL Directive	364
5.24.2	Header Files in HLA	365
5.25	Namespace Pollution	366
5.26	For More Information	369

6

ARITHMETIC

6.1	Chapter Overview	371
6.2	80x86 Integer Arithmetic Instructions	371
6.2.1	The MUL and IMUL Instructions	371
6.2.2	The DIV and IDIV Instructions	375
6.2.3	The CMP Instruction	378
6.2.4	The SETcc Instructions	382
6.2.5	The TEST Instruction	384
6.3	Arithmetic Expressions	385
6.3.1	Simple Assignments	386
6.3.2	Simple Expressions	387
6.3.3	Complex Expressions	389
6.3.4	Commutative Operators	395
6.4	Logical (Boolean) Expressions	396
6.5	Machine and Arithmetic Idioms	398
6.5.1	Multiplying Without MUL, IMUL, or INTMUL	398
6.5.2	Division Without DIV or IDIV	400
6.5.3	Implementing Modulo-N Counters with AND	400
6.5.4	Careless Use of Machine Idioms	401
6.6	Floating Point Arithmetic	401
6.6.1	FPU Registers	402
6.6.2	FPU Data Types	408
6.6.3	The FPU Instruction Set	410
6.6.4	FPU Data Movement Instructions	410
6.6.5	Conversions	412
6.6.6	Arithmetic Instructions	414
6.6.7	Comparison Instructions	420
6.6.8	Constant Instructions	422

6.6.9	Transcendental Instructions	422
6.6.10	Miscellaneous Instructions	424
6.6.11	Integer Operations	426
6.7	Converting Floating Point Expressions to Assembly Language	426
6.7.1	Converting Arithmetic Expressions to Postfix Notation	428
6.7.2	Converting Postfix Notation to Assembly Language	430
6.8	HLA Standard Library Support for Floating Point Arithmetic	431
6.8.1	The <code>stdin.getf</code> and <code>fileio.getf</code> Functions	431
6.8.2	Trigonometric Functions in the HLA Math Library	432
6.8.3	Exponential and Logarithmic Functions in the HLA Math Library	433
6.9	Putting It All Together	434

7

LOW LEVEL CONTROL STRUCTURES

7.1	Chapter Overview	435
7.2	Low Level Control Structures	435
7.3	Statement Labels	436
7.4	Unconditional Transfer of Control (JMP)	438
7.5	The Conditional Jump Instructions	441
7.6	"Medium Level" Control Structures: JT and JF	444
7.7	Implementing Common Control Structures in Assembly Language	445
7.8	Introduction to Decisions	445
7.8.1	IF..THEN..ELSE Sequences	447
7.8.2	Translating HLA IF Statements into Pure Assembly Language	451
7.8.3	Implementing Complex IF Statements Using Complete Boolean Evaluation	456
7.8.4	Short-Circuit Boolean Evaluation	457
7.8.5	Short-Circuit vs. Complete Boolean Evaluation	459
7.8.6	Efficient Implementation of IF Statements in Assembly Language	461
7.8.7	SWITCH/CASE Statements	466
7.9	State Machines and Indirect Jumps	477
7.10	Spaghetti Code	480
7.11	Loops	481
7.11.1	WHILE Loops	482
7.11.2	REPEAT..UNTIL Loops	483
7.11.3	FOREVER..ENDFOR Loops	484
7.11.4	FOR Loops	485
7.11.5	The BREAK and CONTINUE Statements	486
7.11.6	Register Usage and Loops	490
7.12	Performance Improvements	491
7.12.1	Moving the Termination Condition to the End of a Loop	492
7.12.2	Executing the Loop Backward	494
7.12.3	Loop Invariant Computations	495
7.12.4	Unraveling Loops	496
7.12.5	Induction Variables	498
7.13	Hybrid Control Structures in HLA	499
7.14	For More Information	501

8.1	Chapter Overview	503
8.2	File Organization	503
8.2.1	Files as Lists of Records	504
8.2.2	Binary vs. Text Files	506
8.3	Sequential Files	508
8.4	Random Access Files	516
8.5	ISAM (Indexed Sequential Access Method) Files	520
8.6	Truncating a File	524
8.7	For More Information	525

ADVANCED ARITHMETIC

9.1	Chapter Overview	527
9.2	Multiprecision Operations	528
9.2.1	HLA Standard Library Support for Extended Precision Operations	528
9.2.2	Multiprecision Addition Operations	531
9.2.3	Multiprecision Subtraction Operations	534
9.2.4	Extended Precision Comparisons	535
9.2.5	Extended Precision Multiplication	539
9.2.6	Extended Precision Division	543
9.2.7	Extended Precision NEG Operations	553
9.2.8	Extended Precision AND Operations	555
9.2.9	Extended Precision OR Operations	555
9.2.10	Extended Precision XOR Operations	556
9.2.11	Extended Precision NOT Operations	556
9.2.12	Extended Precision Shift Operations	556
9.2.13	Extended Precision Rotate Operations	560
9.2.14	Extended Precision I/O	561
9.3	Operating on Different-Sized Operands	582
9.4	Decimal Arithmetic	584
9.4.1	Literal BCD Constants	585
9.4.2	The 80x86 DAA and DAS Instructions	586
9.4.3	The 80x86 AAA, AAS, AAM, and AAD Instructions	588
9.4.4	Packed Decimal Arithmetic Using the FPU	589
9.5	Tables	591
9.5.1	Function Computation via Table Look-Up	592
9.5.2	Domain Conditioning	597
9.5.3	Generating Tables	598
9.5.4	Table Look-Up Performance	601
9.6	For More Information	602

10

MACROS AND THE HLA COMPILE TIME LANGUAGE

10.1	Chapter Overview	603
10.2	Introduction to the Compile Time Language (CTL)	603
10.3	The #PRINT and #ERROR Statements	605
10.4	Compile Time Constants and Variables	607
10.5	Compile Time Expressions and Operators	607
10.6	Compile Time Functions	610
10.6.1	Type Conversion Compile Time Functions	611
10.6.2	Numeric Compile Time Functions	612
10.6.3	Character Classification Compile Time Functions	613
10.6.4	Compile Time String Functions	613
10.6.5	Compile Time Pattern Matching Functions	614
10.6.6	Compile Time Symbol Information	615
10.6.7	Miscellaneous Compile Time Functions	616
10.6.8	Compile Time Type Conversions of TEXT Objects	617
10.7	Conditional Compilation (Compile Time Decisions)	618
10.8	Repetitive Compilation (Compile Time Loops)	623
10.9	Macros (Compile Time Procedures)	627
10.9.1	Standard Macros	627
10.9.2	Macro Parameters	629
10.9.3	Local Symbols in a Macro	636
10.9.4	Macros as Compile Time Procedures	639
10.9.5	Simulating Function Overloading with Macros	640
10.10	Writing Compile Time "Programs"	646
10.10.1	Constructing Data Tables at Compile Time	646
10.10.2	Unrolling Loops	651
10.11	Using Macros in Different Source Files	653
10.12	For More Information	653

11

BIT MANIPULATION

11.1	Chapter Overview	655
11.2	What Is Bit Data, Anyway?	656
11.3	Instructions That Manipulate Bits	657
11.4	The Carry Flag as a Bit Accumulator	665
11.5	Packing and Unpacking Bit Strings	666
11.6	Coalescing Bit Sets and Distributing Bit Strings	669
11.7	Packed Arrays of Bit Strings	671
11.8	Searching for a Bit	673
11.9	Counting Bits	676
11.10	Reversing a Bit String	679
11.11	Merging Bit Strings	681
11.12	Extracting Bit Strings	682
11.13	Searching for a Bit Pattern	683
11.14	The HLA Standard Library Bits Module	684
11.15	For More Information	687

12

THE STRING INSTRUCTIONS

12.1	Chapter Overview	689
12.2	The 80x86 String Instructions	690
12.2.1	How the String Instructions Operate	690
12.2.2	The REP/REPE/REPZ and REPNZ/REPNE Prefixes	691
12.2.3	The Direction Flag	692
12.2.4	The MOVS Instruction	694
12.2.5	The CMPS Instruction	700
12.2.6	The SCAS Instruction	703
12.2.7	The STOS Instruction	704
12.2.8	The LODS Instruction	705
12.2.9	Building Complex String Functions from LODS and STOS	705
12.3	Performance of the 80x86 String Instructions	706
12.4	For More Information	707

13

THE MMX INSTRUCTION SET

13.1	Chapter Overview	709
13.2	Determining Whether a CPU Supports the MMX Instruction Set	710
13.3	The MMX Programming Environment	711
13.3.1	The MMX Registers	711
13.3.2	The MMX Data Types	713
13.4	The Purpose of the MMX Instruction Set	714
13.5	Saturation Arithmetic and Wrap-Around Mode	714
13.6	MMX Instruction Operands	715
13.7	MMX Technology Instructions	717
13.7.1	MMX Data Transfer Instructions	718
13.7.2	MMX Conversion Instructions	718
13.7.3	MMX Packed Arithmetic Instructions	723
13.7.4	MMX Logical Instructions	726
13.7.5	MMX Comparison Instructions	727
13.7.6	MMX Shift Instructions	731
13.7.7	The EMMS Instruction	733
13.8	The MMX Programming Paradigm	734
13.9	For More Information	745

14

CLASSES AND OBJECTS

14.1	Chapter Overview	747
14.2	General Principles	748
14.3	Classes in HLA	750
14.4	Objects	753
14.5	Inheritance	755

14.6	Overriding	756
14.7	Virtual Methods vs. Static Procedures	757
14.8	Writing Class Methods and Procedures	759
14.9	Object Implementation	764
14.9.1	Virtual Method Tables	767
14.9.2	Object Representation with Inheritance	769
14.10	Constructors and Object Initialization	773
14.10.1	Dynamic Object Allocation Within the Constructor	775
14.10.2	Constructors and Inheritance	777
14.10.3	Constructor Parameters and Procedure Overloading	781
14.11	Destructors	782
14.12	HLA's "_initialize_" and "_finalize_" Strings	783
14.13	Abstract Methods	789
14.14	Run-Time Type Information (RTTI)	792
14.15	Calling Base Class Methods	794
14.16	For More Information	795

15

MIXED LANGUAGE PROGRAMMING

15.1	Chapter Overview	797
15.2	Mixing HLA and MASM/Gas Code in the Same Program	798
15.2.1	In-Line (MASM/Gas) Assembly Code in Your HLA Programs	798
15.2.2	Linking MASM/Gas-Assembled Modules with HLA Modules	801
15.3	Programming in Delphi/Kylix and HLA	805
15.3.1	Linking HLA Modules with Delphi/Kylix Programs	806
15.3.2	Register Preservation	810
15.3.3	Function Results	811
15.3.4	Calling Conventions	817
15.3.5	Pass by Value, Reference, CONST, and OUT in Kylix	823
15.3.6	Scalar Data Type Correspondence Between Delphi/Kylix and HLA ..	825
15.3.7	Passing String Data Between Delphi/Kylix and HLA Code	826
15.3.8	Passing Record Data Between HLA and Kylix	829
15.3.9	Passing Set Data Between Delphi/Kylix and HLA	833
15.3.10	Passing Array Data Between HLA and Delphi/Kylix	834
15.3.11	Referencing Delphi/Kylix Objects from HLA Code	834
15.4	Programming in C/C++ and HLA	837
15.4.1	Linking HLA Modules with C/C++ Programs	839
15.4.2	Register Preservation	842
15.4.3	Function Results	842
15.4.4	Calling Conventions	842
15.4.5	Pass by Value and Reference in C/C++	847
15.4.6	Scalar Data Type Correspondence Between C/C++ and HLA	847
15.4.7	Passing String Data Between C/C++ and HLA Code	849
15.4.8	Passing Record/Structure Data Between HLA and C/C++	849
15.4.9	Passing Array Data Between HLA and C/C++	851
15.5	For More Information	852

A
ASCII CHARACTER SET
853

B
THE 80X86 INSTRUCTION SET
857

INDEX
889

**HELLO, WORLD
OF ASSEMBLY LANGUAGE**



1.1 Chapter Overview

This chapter is a "quick start" chapter that lets you start writing basic assembly language programs as rapidly as possible. This chapter:

- Presents the basic syntax of an HLA (High Level Assembly) program
- Introduces you to the Intel CPU architecture
- Provides a handful of data declarations, machine instructions, and high level control statements
- Describes some utility routines you can call in the HLA Standard Library
- Shows you how to write some simple assembly language programs

By the conclusion of this chapter, you should understand the basic syntax of an HLA program and should understand the prerequisites that are needed to start learning new assembly language features in the chapters that follow.